Санкт-Петербургский государственный политехнический университет

Кафедра компьютерных систем и программных технологий

**Отчёт лабораторной работе №6**

**Дисциплина**: Базы данных

**Тема**: Изучение работы транзакций

Выполнил студент гр. 43501/1 А.М. Зинченко

(подпись)

Руководитель А.В. Мяснов

(подпись)

“ ” 2014 г.

Санкт - Петербург

2014

1. **Цель работы**

Изучить механизм транзакций, возможности ручного управления транзакциями, уровни изоляции транзакций.

1. **Проведение работы**

*Провести эксперименты по запуску, подтверждению и откату транзакций*

Добавим в таблицу Vagons\_types, несколько новых типов:

insert into VAGONS\_TYPES (VAGONS\_TYPES\_ID, VAGONS\_TYPES\_NAMES, NUMB\_OF\_PLACES)values (4, 'Люкс-2', 100);

insert into VAGONS\_TYPES (VAGONS\_TYPES\_ID, VAGONS\_TYPES\_NAMES, NUMB\_OF\_PLACES)values (5, 'Люкс-3', 100);

select \* from vagons\_types;

VAGONS\_TYPES\_ID VAGONS\_TYPES\_NAMES NUMB\_OF\_PLACES

=============== ================== ==============

1 Сидячий-3 100

2 Плацкарт 54

3 Люкс 30

5 Люкс-3 100

4 Люкс-2 100

Откроем еще один сеанс с БД и выполним запрос:

set transaction;

select \* from vagons\_types;

VAGONS\_TYPES\_ID VAGONS\_TYPES\_NAMES NUMB\_OF\_PLACES

=============== ================== ==============

1 Сидячий-3 100

2 Плацкарт 54

3 Люкс 30

При откате транзакции в обоих сеансах действия по добавлению записей будут отменены.

SQL> rollback;

SQL> select \* from vagons\_types;

VAGONS\_TYPES\_ID VAGONS\_TYPES\_NAMES NUMB\_OF\_PLACES

=============== ================== ==============

1 Сидячий-3 100

2 Плацкарт 54

3 Люкс 30

Снова добавим те же данные, но с подтверждением в первом и посмотрим на результат во втором сеансе:

commit;

Во втором:

set transaction;

select \* from vagons\_types;

VAGONS\_TYPES\_ID VAGONS\_TYPES\_NAMES NUMB\_OF\_PLACES

=============== ================== ==============

1 Сидячий-3 100

2 Плацкарт 54

3 Люкс 30

5 Люкс-3 100

4 Люкс-2 100

*Разобраться с уровнями изоляции транзакций в Firebird.Спланировать и провести эксперименты, показывающие основные возможности транзакций с различным уровнем изоляции.*

Уровень изолированности транзакции определяет, какие изменения, сделанные в других транзакциях, будут видны в данной транзакции. Каждая транзакция имеет свой уровень изоляции, который устанавливается при ее запуске и остается неизменным в течение всей ее жизни.

Транзакции в Firebird могут иметь 3 основных возможных уровня изоляции: READ COMMITTED, SNAPSHOT и SNAPSHOT TABLE STABILITY. Каждый из этих трех уровней изоляции определяет правила видимости тех действий, которые выполняются другими транзакциями.

• READ COMMITTED ( "читать подтвержденные данные"). Уровень изоляции READCOMMITTED используется, когда мы хотим видеть все подтвержденные результаты параллельно выполняющихся (т. е. в рамках других транзакций) действий. Этот уровень изоляции гарантирует, что мы не сможем прочитать неподтвержденные данные, измененные в других транзакциях, и делает

возможным прочитать подтвержденные данные.

• SNAPSHOT. Этот уровень изоляции используется для создания "моментального" снимка базы данных. Все операции чтения данных, выполняемые в рамках транзакции с уровнем изоляции SNAPSHOT, будут видеть только состояние базы данных на момент начала запуска транзакции. Все изменения, сделанные в параллельных транзакциях, не видны в этой транзакции. В то же время SNAPSHOT не блокирует данные, которые он не изменяет.

• SNAPSHOT TABLE STABILITY. Это уровень изоляции также создает "моментальный" снимок базы данных, но одновременно блокирует на запись данные, задействованные в операциях, выполняемые данной транзакцией. Это означает, что если транзакция SNAPSHOT TABLE STABILITY изменила данные в какой-нибудь таблице, то после этого данные в этой таблице уже не могут быть изменены в других параллельных транзакциях. Кроме того, транзакции с уровнем изоляции SNAPSHOT TABLE STABILITY не могут получить доступ к таблице, если данные в них уже изменяются в контексте других транзакций.

**Уровень изоляции READ COMMITTED**

Эксперимент:

1. set transaction isolation level read committed;

2. insert into VAGONS\_TYPES (VAGONS\_TYPES\_ID, VAGONS\_TYPES\_NAMES, NUMB\_OF\_PLACES)values (4, 'Lux-2', 100);

1. select \* from vagons\_types;

1. Deadlock

2. Commit;

1. Чтение произошло

1 сессия:

SQL> set transaction isolation level read committe

Commit current transaction (y/n)?y

Committing.

SQL> select \* from vagons\_types;

VAGONS\_TYPES\_ID VAGONS\_TYPES\_NAMES NUMB\_OF\_PLACES

=============== ================== ==============

1 Сидячий-3 100

2 Плацкарт 54

3 Люкс 30

4 Lux-2 100

2 сессия:

SQL> insert into VAGONS\_TYPES (VAGONS\_TYPES\_ID, VAGONS\_TYPES\_NAMES, NUMB\_OF\_

PLACES)values (4, 'Lux-2', 100);

SQL> commit;

Уровень изоляции READ COMMITTED имеет два режима - NO RECORD VERSION и RECORD VERSION. По умолчанию включен первый режим. Проведем тот же эксперимент включив второй режим:

1 сессия:

SQL> set transaction isolation level read committed RECORD\_VERSION;

SQL> select \* from vagons\_types;

VAGONS\_TYPES\_ID VAGONS\_TYPES\_NAMES NUMB\_OF\_PLACES

=============== ================== ==============

1 Сидячий-3 100

2 Плацкарт 54

3 Люкс 30

2 сессия:

SQL> insert into VAGONS\_TYPES (VAGONS\_TYPES\_ID, VAGONS\_TYPES\_NAMES, NUMB\_OF\_PLACES) values (4, 'Lux-2', 100);

**Уровень изоляции SNAPSHOT**

Эксперимент:

1. set transaction isolation level snapshot;

2. insert into VAGONS\_TYPES (VAGONS\_TYPES\_ID, VAGONS\_TYPES\_NAMES, NUMB\_OF\_PLACES)values (4, 'Lux-2', 100);

2. commit;

1. select \* from vagons\_types;

1 сессия:

SQL> set transaction isolation level snapshot;

Commit current transaction (y/n)?y

Committing.

SQL> select \* from vagons\_types;

VAGONS\_TYPES\_ID VAGONS\_TYPES\_NAMES NUMB\_OF\_PLACES

=============== ================== ==============

1 Сидячий-3 100

2 Плацкарт 54

3 Люкс 30

2 сессия:

SQL> insert into VAGONS\_TYPES (VAGONS\_TYPES\_ID, VAGONS\_TYPES\_NAMES,NUMB\_OF\_PLACES)values (4, 'Lux-2', 100);

SQL> commit;

Эксперимент 2:

1. set transaction isolation level snapshot;

2. update vagons\_types set numb\_of\_places=10 where vagons\_types\_id=4;

1. update vagons\_types set numb\_of\_places=15 where vagons\_types\_id=4;

1. Deadlock

2. commit;

1 сессия:

SQL> set transaction isolation level snapshot;

Commit current transaction (y/n)?y

Committing.

SQL> select \* from vagons\_types;

VAGONS\_TYPES\_ID VAGONS\_TYPES\_NAMES NUMB\_OF\_PLACES

=============== ================== ==============

1 Сидячий-3 100

2 Плацкарт 54

3 Люкс 30

4 Lux-2 100

SQL> update vagons\_types set numb\_of\_places=15 where vagons\_types\_id=4;

Statement failed, SQLSTATE = 40001

deadlock

-update conflicts with concurrent update

-concurrent transaction number is 14960

2 сессия:

SQL> update vagons\_types set numb\_of\_places=10 where vagons\_types\_id=4;

SQL> commit;

**Уровень изоляции SNAPSHOT TABLE STABILITY**

Эксперимент:

1. set transaction isolation level snapshot TABLE STABILITY;

2. set transaction isolation level snapshot TABLE STABILITY;

1. update vagons\_types set numb\_of\_places=10 where vagons\_types\_id=4;

2. select \* from vagons\_types

2. Deadlock

1. Commit;

1. select \* from vagons\_types;

2. select \* from vagons\_types;

1 сессия:

SQL> set transaction isolation level snapshot TABLE STABILITY;

SQL> update vagons\_types set numb\_of\_places=25 where vagons\_types\_id=4;

SQL> commit;

SQL> select \* from vagons\_types;

VAGONS\_TYPES\_ID VAGONS\_TYPES\_NAMES NUMB\_OF\_PLACES

=============== ================== ==============

1 Сидячий-3 100

2 Плацкарт 54

3 Люкс 30

4 Lux-2 25

2 сессия:

SQL> set transaction isolation level snapshot TABLE STABILITY;

Commit current transaction (y/n)?y

Committing.

SQL> select \* from vagons\_types;

VAGONS\_TYPES\_ID VAGONS\_TYPES\_NAMES NUMB\_OF\_PLACES

=============== ================== ==============

1 Сидячий-3 100

2 Плацкарт 54

3 Люкс 30

4 Lux-2 10

SQL> select \* from vagons\_types;

VAGONS\_TYPES\_ID VAGONS\_TYPES\_NAMES NUMB\_OF\_PLACES

=============== ================== ==============

1 Сидячий-3 100

2 Плацкарт 54

3 Люкс 30

4 Lux-2 10

Основное отличие SNAPSHOT TABLE STABILITY от SNAPSHOT в том, что таблица была заблокирована и на чтение.

1. **Выводы**

Транзакция — группа последовательных операций с базой данных, которая представляет собой логическую единицу работы с данными. Транзакция может быть выполнена либо целиком и успешно, соблюдая целостность данных и независимо от параллельно идущих других транзакций, либо не выполнена вообще и тогда она не должна произвести никакого эффекта. Одним из наиболее распространённых наборов требований к транзакциям и транзакционным системам является набор ACID (Atomicity, Consistency, Isolation, Durability).

Для поддержания поставленных требований и эффективного баланса производительности в СУБД введены различные системы изолированности транзакций. Ниже рассмотрено соответствие между уровнем изоляции и режимом транзакции:

|  |  |
| --- | --- |
| Уровни | Режим |
| 0 — Чтение неподтверждённых данных (грязное чтение) | Нет |
| 1 — Чтение подтверждённых данных | READ COMMITTED |
| 2 — Повторяемое чтение | SNAPSHOT |
| 3 — Сериализуемый | SNAPSHOT TABLE STABILITY |